View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Elsevier - Publisher Connector

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER Electronic Notes in Theoretical Computer Science 91 (2004) 171–194
www.elsevier.com/locate/entcs

A Membership Algorithm for Functional and Multi-valued Dependencies in the Presence of Lists

Sven Hartmann[1] and Sebastian Link[2]

Information Science Research Centre,
Massey University,
Palmerston North, New Zealand

Abstract

Nested lists are used as a data structure whenever order matters. List types are therefore supported by many advanced data models such as genomic sequence, deductive and object-oriented data models including XML.
What impact does the finite list type have on the two most important classes of relational dependencies? The membership problem of functional and multi-valued dependencies in databases supporting base, record and list types is investigated. The problem is to decide whether a functional or multi-valued dependency follows from a given set of functional and multi-valued dependencies. In order to capture different data models at a time, an abstract algebraic approach based on nested attributes and subtyping is taken. This algebraic framework allows to generalise Beeri's well-known membership algorithm in [6] from the relational data model. It is argued that the algorithm presented works correctly and in polynomial time.

Keywords: Advanced Data Models, Dependencies, Finite Implication Problem, Correctness, Complexity

1 Introduction

1.1 Dependency Theory in Relational Databases

In designing databases the semantics of the application domain has to captured as completely as possible. As this cannot be expressed solely by structures,

[1] Email:s.hartmann@massey.ac.nz
[2] Email:s.link@massey.ac.nz

1571-0661 © 2004 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2003.12.012

we have to use dependencies, i.e., sentences in a logic suitable for the data model used. Database theory has to investigate the implications arising from the presence of dependencies. This means to describe semantically desirable properties of "well-designed" databases, e.g., the absence of redundancy, to characterise (if possible) them syntactically by in-depth investigation of the dependencies and to develop algorithms to transform schemata into normal forms, which guarantee the desirable properties to be satisfied.

In the relational data model (RDM, [2,36]) a lot of research has been done on dependency theory and normal forms. Starting with the seminal work by Codd [20] normal forms such as third normal form (3NF), Boyce-Codd normal form (BCNF, [13,21]) and fourth normal form (4NF, [22,23,24]) have been introduced to characterise the absence of redundancy and update anomalies in the presence of functional and multi-valued dependencies (FDs, MVDs), though a theoretically convincing justification for these normal forms was given only 20 years later [45]. Roughly speaking, a functional dependency $X \to Y$ requires that whenever two tuples of a relation coincide on X, they must also coincide on Y. A multi-valued dependency $X \twoheadrightarrow Y$ requires that whenever two tuples of a relation coincide on X, their values on Y must be mutually exchangeable and thus generate additional tuples.

Various other classes of dependencies for the RDM have been introduced (see [41] for an overview) and large parts of database theory deals with the finite axiomatisation of these dependencies and the finite implication problem for them, i.e., to decide that a dependency φ is implied by a set of dependencies Σ, where implication refers to the fact that all finite models of Σ are also models of φ. Armstrong [4] was the first to give a finite axiomatisation for FDs, and Beeri and others gave a finite axiomatisation for FDs and MVDs [9] and developed various versions of efficient decision algorithms [6,7,8].

1.2 Challenges in Advanced Data Models

The need to store data beyond relational structure has become more and more apparent over the years. Many new and different data models have been introduced. First, so called semantic data models have been developed [18,30], which were originally just meant to be used as design aids, as application semantics was assumed to be easier captured by these models [5,19,44]. Later on some of these models, especially the nested relational model [36], object oriented models [38] and object-relational models, the gist of which are captured by the higher-order Entity-Relationship model (HERM, [42,43]) have become interesting as data models in their own right.

One key problem is to develop dependency theories (or preferably a unified theory) for the most relevant advanced data models. These are probably

the HERM as a nested model with various bulk type constructors, good theoretical foundations and proven practical relevance [43], the object oriented model [38], the semi-structured data model and XML [1], which add unions and most importantly references, the expansion of which leads to rational tree structures. The development of such a dependency theory will have a significant impact on understanding application semantics and laying the grounds for a logically founded theory of well-designed databases.

Biskup [15,16] lists in particular two challenges for database design theory: finding a unifying framework and extending achievements to deal with advanced database features such as complex object types.

1.3 Contributions

This paper continues to take on these two challenges. In order to find a unifying framework and capture several data models at a time our work is based on an abstract approach in the context of types for nested attributes and subtyping. Recall that in the RDM, the set of all subsets of a relation schema forms a Boolean Algebra with respect to inclusion and the set operations of union, intersection and difference. This simple, yet powerful tool is the basis for the success of relational dependency theory. We follow a similar algebraic approach in order to extend these achievements to complex object types.

This article considers FDs and MVDs in the presence of record and list types. There are several reasons for the importance of lists. First of all, lists are perhaps the most common data type. The need for lists arises from applications that store ordered relations, time-series data, meteorological and astronomical data streams, runs of experimental data, multidimenstional arrays, textual information, voices, sound, images, video, etc. Lists have been subject to studies in the deductive and temporal database community for some time [37,34]. The list type also naturally appears in object-oriented databases [38,25] and is in particular important for XML [1,47]. Recently, bioinformatics has become a very important field of research. Of course, lists occur naturally in genomic sequence databases [39,17].

The main result in [29] presents a finite axiomatisation of FDs and MVDs in the presence of record and finite list types extending the work in [9] from the RDM. This paper presents a membership algorithm for deciding the finite implication problem of FDs and MVDs. Due to the natural algebraic approach, this algorithm is quite similar to Beeri's membership algorithm from the RDM ([6]). Such an algorithm for deciding implication of dependencies can be used to decide the equivalence of two sets of dependencies or the redundancy of a given set of dependencies. This is considered a significant step towards automated database schema design [12,10,11] which some researchers

see as the ultimate goal for research in dependency theory [8].

Furthermore, we formally demonstrate that the algorithm works correctly and in polynomial time. Finally, the paper shows that the algebraic approach based on nested attributes seems to be very natural. That is, established results from relational database theory can be carried over to complex object databases. The nesting of attributes can be easily extended to sets, multisets, unions, references etc. Therefore, our work might serve as a unifying framework for the foundation of dependency theory in advanced data models.

1.4 Outline

The paper is structured as follows. Section 2 briefly comments on related work. After introducing fundamental definitions in Section 3, it is demonstrated that the set of subattributes for some fixed nested attribute carries the structure of a Brouwerian Algebra (co-Heyting Algebra). This generalises the framework of a Boolean Algebra from the RDM. We repeat in Section 4 how to obtain a sound and complete set of inference rules for the implication of FDs and MVDs in the presence of base, record and list types (see [29]). The inference rules presented are natural generalisations of their counterparts from relational databases. It turns out, however, that an additional rule allowing the derivation of non-trivial FDs from MVDs is needed (which is impossible in the RDM). In order to decide the membership problem it is sufficient to compute the dependency basis of some nested attribute. Section 5 presents an algorithm for computing the dependency basis. The algorithm is a natural extension of Beeri's membership algorithm from the RDM ([6]). The correctness proof, outlined in Section 6, is based on the finite axiomatisation from [29] using the algebraic framework. Finally, Section 6 demonstrates that the membership problem can be solved in polynomial time. We conclude in Section 7 and comment on future work.

2 Previous and Related Work

The work in [33] and [35] study normalisation in the nested relational data model where nesting refers to sets rather than lists. Both approaches define FDs based on the notion of a path and do not derive any axiomatisation results nor algorithms for deciding the (finite) implication problem. We have also studied FDs in the presence of the finite set type and provided a finite axiomatisation [27] and a normal form proposal [28] proving the equivalence to the absence of redundancy and sufficiency for the absence of any update anomalies in the spirit of [45]. The expressiveness of FDs, based on our algebraic approach in the presence of sets only, deviates from those in either

works mentioned above.

The article [40] considers normalisation in an object-oriented framework. The authors provide an extension of functional dependencies to cope with the richer semantics of relationships between objects, called path dependency, local dependency, and global dependency constraints. An object is said to be in normal form if and only if the user's interpretation of this object is derivable from the model of the object. Again, this approach is very different from ours.

The works in [3] and [46] define FDs in the context of XML. Both notions are again based on paths. [3] derives some complexity results for the implication problem of certain subclasses of FDs and certain DTDs. The only paper that discusses MVDs is [46]. There are, however, no results on axiomatisation nor any discussion of the implication problem. In fact, the approaches are entirely different from ours. Note that there are different notions of FDs in the context of XML which all lead to a different expressiveness. See [26] for a discussion. Normalisation should then be studied from different points of view.

The approach in this paper specifically focuses on lists and is of algebraic nature. The authors are not aware of any similar work in the literature.

3 The Algebra of Nested Attributes

This section introduces a data model based on the nesting of attributes and subtyping. It may be used to provide a unifying framework for the study of complex object types such as records, lists, sets, multisets, unions and references. This article, however, focuses on records and lists only.

3.1 Nested Attributes

We start with the definition of flat attributes and values for them.

Definition 3.1 A *universe* is a finite set $\mathcal{U}$ together with domains (, i.e., sets of values) $dom(A)$ for all $A \in \mathcal{U}$. The elements of $\mathcal{U}$ are called *flat attributes*.□

For the relational data model a universe was sufficient. That is, a relation schema is defined by a finite and non-empty subset $\mathcal{R} \subseteq \mathcal{U}$. For higher-order data models, however, nested attributes are needed. In the following definition we use a set $\mathcal{L}$ of labels, and assume that the symbol λ is neither a flat attribute nor a label, i.e., $\lambda \notin \mathcal{U} \cup \mathcal{L}$. Moreover, flat attributes are not labels and vice versa, i.e., $\mathcal{U} \cap \mathcal{L} = \emptyset$.

Definition 3.2 Let $\mathcal{U}$ be a universe and $\mathcal{L}$ a set of labels. The set $\mathcal{N}\mathrm{A} = \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ of *nested attributes over* $\mathcal{U}$ *and* $\mathcal{L}$ is the smallest set satisfying the following conditions:

- $\lambda \in \mathcal{N}\mathrm{A}$,
- $\mathcal{U} \subseteq \mathcal{N}\mathrm{A}$,
- for $L \in \mathcal{L}$ and $N_1, \ldots, N_k \in \mathcal{N}\mathrm{A}$ with $k \geq 1$ we have $L(N_1, \ldots, N_k) \in \mathcal{N}\mathrm{A}$,
- for $L \in \mathcal{L}$ and $N \in \mathcal{N}\mathrm{A}$ we have $L[N] \in \mathcal{N}\mathrm{A}$.

We call λ *null attribute*, $L(N_1, \ldots, N_k)$ *record-valued attribute* and $L[N]$ *list-valued attribute*. □

We can now extend the mapping *dom* from flat attributes to nested attributes, i.e., we define a set $dom(N)$ of values for every nested attribute $N \in \mathcal{N}\mathrm{A}$.

Definition 3.3 For a nested attribute $N \in \mathcal{N}\mathrm{A}$ we define the *domain* $dom(N)$ as follows:

- $dom(\lambda) = \{ok\}$,
- $dom(L(N_1, \ldots, N_k)) = \{(v_1, \ldots, v_k) \mid v_i \in dom(N_i) \text{ for } i = 1, \ldots, k\}$, i.e., the set of all k-tuples $(v_1, \ldots, v_k)$ with $v_i \in dom(N_i)$ for all $i = 1, \ldots, k$, and
- $dom(L[N]) = \{[v_1, \ldots, v_n] \mid v_i \in dom(N) \text{ for } i = 1, \ldots, n\}$, i.e., the set of all finite lists with elements in $dom(N)$. □

The empty list is denoted by [ ]. Note that the relational data model is completely covered by the presence of tuple-valued attributes only. Instead of relation schemata R we will now consider a nested attribute N, assuming that a universe $\mathcal{U}$ and a set $\mathcal{L}$ of labels are fixed. An R-relation r is then replaced by some finite set $r \subseteq dom(N)$.

3.2 Subattributes

Dependency theory in the relational data model is based on the powerset $\mathcal{P}(R)$ for a relation schema R. In fact, $\mathcal{P}(R)$ is a powerset algebra with partial order $\subseteq$, set union $\cup$, set intersection $\cap$ and set difference $-$. We will generalise these operations for nested attributes starting with a partial order $\leq$.

Definition 3.4 The *subattribute relation* $\leq$ on the set of nested attributes $\mathcal{N}\mathrm{A}$ over $\mathcal{U}$ and $\mathcal{L}$ is defined by the following rules, and the following rules only:

- $N \leq N$ for all nested attributes $N \in \mathcal{N}\mathrm{A}$,
- $\lambda \leq A$ for all flat attributes $A \in \mathcal{U}$,
- $\lambda \leq N$ for all list-valued attributes $N \in \mathcal{N}\mathrm{A}$,
- $L(N_1, \ldots, N_k) \leq L(M_1, \ldots, M_k)$ whenever $N_i \leq M_i$ for all $i = 1, \ldots, k$,
- $L[N] \leq L[M]$ whenever $N \leq M$.

For $N, M \in \mathcal{N}\mathrm{A}$ we say that M is a *subattribute* of N if and only if $M \leq N$ holds. We write $M \not\leq N$ if and only if M is not a subattribute of N. □

The subattribute relation $\leq$ on nested attributes is reflexive, anti-symmetric and transitive.

Lemma 3.5 *The subattribute relation is a partial order on nested attributes.*□

Informally, $M \leq N$ for $N, M \in \mathcal{N}\mathrm{A}$ if and only if M comprises at most as much information as N does. The informal description of the subattribute relation is formally documented by the existence of a projection function $\pi^N_M : Dom(N) \to Dom(M)$ in case $M \leq N$ holds.

Definition 3.6 Let $N, M \in \mathcal{N}\mathrm{A}$ with $M \leq N$. The *projection function* $\pi^N_M : Dom(N) \to Dom(M)$ is defined as follows:

- if $N = M$, then $\pi^N_M = id_{Dom(N)}$ is the identity on $dom(N)$,
- if $M = \lambda$, then $\pi^N_\lambda : Dom(N) \to \{ok\}$ is the constant function that maps every $v \in Dom(N)$ to ok,
- if $N = L(N_1, \ldots, N_k)$ and $M = L(M_1, \ldots, M_k)$, then $\pi^N_M = \pi^{N_1}_{M_1} \times \cdots \times \pi^{N_k}_{M_k}$ which maps every tuple $(v_1, \ldots, v_k) \in Dom(N)$ to $(\pi^{N_1}_{M_1}(v_1), \ldots, \pi^{N_k}_{M_k}(v_k)) \in Dom(M)$, and
- if $N = L[N']$ and $M = L[M']$, then $\pi^N_M : Dom(N) \to Dom(M)$ maps every list $[v_1, \ldots, v_n] \in Dom(N)$ to the list $[\pi^{N'}_{M'}(v_1), \ldots, \pi^{N'}_{M'}(v_n)] \in Dom(M)$. □

Let $\mathcal{X}, \mathcal{Y}$ be two sets of nested attributes. $\mathcal{X}$ is called a *generalised subset* of $\mathcal{Y}$, denoted by $\mathcal{X} \subseteq_{\mathrm{gen}} \mathcal{Y}$ if and only if for every $X \in \mathcal{X}$ there is some $Y \in \mathcal{Y}$ with $X \leq Y$. Note that $\subseteq_{\mathrm{gen}}$ is a pre-order on sets of nested attributes.

3.3 The Brouwerian Algebra of Subattributes

Fix a set $\mathcal{U}$ of attribute names, and a set $\mathcal{L}$ of labels.

Definition 3.7 Let $N \in \mathcal{N}\mathrm{A}$ be a nested attribute. The set $Sub(N)$ of *subattributes* of N is $Sub(N) = \{M \mid M \leq N\}$. The *bottom element* λ_N of $Sub(N)$ is given by $\lambda_N = L(\lambda_{N_1}, \ldots, \lambda_{N_k})$ whenever $N = L(N_1, \ldots, N_k)$, and $\lambda_N = \lambda$ whenever N is not a tuple-valued attribute. □

We study the algebraic structure of $Sub(N)$. A *Brouwerian Algebra* [31] is a lattice $(L, \sqsubseteq, \sqcup, \sqcap, \dot{-}, 1)$ with top element 1 and a binary operation $\dot{-}$ which satisfies $a \dot{-} b \sqsubseteq c$ iff $a \sqsubseteq b \sqcup c$ for all $c \in L$. In this case, the operation $\dot{-}$ is called the *pseudo-difference*. The *Brouwerian complement* $\neg a$ of $a \in L$ is then defined by $\neg a = 1 \dot{-} a$. A Brouwerian Algebra is also called a co-Heyting Algebra or a dual Heyting Algebra. The system of all closed subsets of a topological space is a well-known Brouwerian Algebra. It is obvious that

$(Sub(N), \leq, \lambda_N, N)$ is a partially ordered set with bottom element λ_N and top element N.

Definition 3.8 Let $N \in \mathcal{NA}$ and $Y, Z \in Sub(N)$. The *join* $Y \sqcup_N Z$, *meet* $Y \sqcap_N Z$ and *pseudo difference* $Y \dot{-}_N Z$ of Y and Z in $Sub(N)$ are inductively defined as follows:

- $Y \sqcup_N Z = Z$ iff $Y \leq Z$ iff $Y \sqcap_N Z = Y$ and $Z \dot{-}_N \lambda_N = Z$, and $Z \leq Y$ iff $Z \dot{-}_N Y = \lambda_N$,
- if $N = L[M]$, $Y = L[A], Z = L[B]$, then $Y \circ_N Z = L[A \circ_M B]$ for $\circ \in \{\sqcup, \sqcap\}$ and if $Z \not\leq Y$, then $Z \dot{-}_N Y = L[B \dot{-}_M A]$.
- if $N = L(N_1, \ldots, N_n), Y = L(A_1, \ldots, A_n)$ and $Z = L(B_1, \ldots, B_n)$, then $Y \circ_N Z = L(A_1 \circ_{N_1} B_1, \ldots, A_n \circ_{N_n} B_n)$ for $\circ \in \{\sqcup, \sqcap, \dot{-}\}$. □

In order to simplify notation, occurences of λ in a tuple-valued attribute are usually omitted if this does not cause any ambiguities. That is, the subattribute $L(M_1, \ldots, M_k) \leq L(N_1, \ldots, N_k)$ is abbreviated by $L(M_{i_1}, \ldots, M_{i_l})$ where $\{M_{i_1}, \ldots, M_{i_l}\} = \{M_j : M_j \neq \lambda_{N_j} \text{ and } 1 \leq j \leq k\}$ and $i_1 < \cdots < i_l$. If $M_j = \lambda_{N_j}$ for all $j = 1, \ldots, k$, then we use λ instead of $L(M_1, \ldots, M_k)$. The subattribute $L_1(A, \lambda, L_2[L_3(\lambda, \lambda)])$ of $L_1(A, B, L_2[L_3(C, D)])$ is abbreviated by $L_1(A, L_2[\lambda])$. However, the subattribute $L(A, \lambda)$ of $L(A, A)$ cannot be abbreviated by $L(A)$ since this may also refer to $L(\lambda, A)$.

If the context allows, we omit the index N from the operations $\sqcup_N, \sqcap_N, \dot{-}_N$ and from λ_N. The Brouwerian Algebra for $J[K(A, L[M(B, C)])]$ is illustrated in Figure 1.

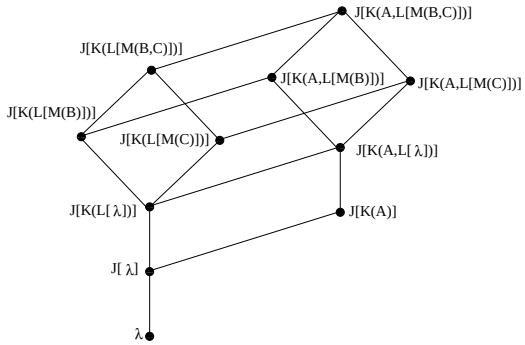

Fig. 1. The Brouwerian Algebra of $J[K(A, L[M(B, C)])]$

Given some nested attribute $N \in \mathcal{NA}$ and $Y, Z \in Sub(N)$, we use $Y_N^{\mathcal{C}} = N \dot{-} Y$ to denote the *Brouwerian complement* of Y in $Sub(N)$. Again, we omit the subscript N if the context allows. The pseudo difference $Z \dot{-} Y$ of Z and Y in

$Sub(N)$ satisfies

$$Z \dot{-} Y \leq X \quad \text{if and only if} \quad Z \leq Y \sqcup X$$

for all $X \in Sub(N)$. Consequently, for all $X \in Sub(N)$ holds $Y^{\mathcal{C}} \leq X$ if and only if $X \sqcup Y = N$ holds.

The following result is straightforward to see: $Sub(\lambda)$ is isomorphic to the Boolean Algebra of order 0, $Sub(A)$, A a flat attribute, isomorphic to the Boolean Algebra of order 1. $Sub(L(P))$ is isomorphic to $Sub(P)$, $Sub(L(P_1, \ldots, P_n))$ isomorphic to the direct product of $Sub(P_1), \ldots, Sub(P_n)$, and $Sub(L[P])$ is isomorphic to $Sub(P)$ augmented by a new minimum. It is an easy exercise to show that the set of all (finite) Brouwerian Algebras is closed with respect to both operations (add a new minimum, direct product). The following theorem generalises the fact that $(\mathcal{P}(R), \subseteq, \cup, \cap, -, \emptyset, R)$ is a Boolean Algebra for a relation schema R in the RDM.

Theorem 3.9 $(Sub(N), \leq, \sqcup_N, \sqcap_N, \dot{-}_N, N)$ *forms a Brouwerian Algebra for every* $N \in \mathcal{N}\mathrm{A}$. □

Note that $(Sub(N), \leq, \sqcup, \sqcap, (\cdot)^{\mathcal{C}}, \lambda, N)$ is in general not boolean. Take for instance $N = L[A]$ and $Y = L[\lambda]$. Then $Y^{\mathcal{C}} = N$ and $Y \sqcap Y^{\mathcal{C}} = Y \neq \lambda$. Furthermore, $Y^{\mathcal{CC}} = \lambda \neq Y$. Moreover, every Brouwerian Algebra is distributive.

4 Axiomatising FDs and MVDs

Definition 4.1 Let $N \in \mathcal{N}\mathrm{A}$ be a nested attribute. A *functional dependency on* N is an expression of the form $X \to Y$ where $X, Y \in Sub(N)$. A finite set $r \subseteq Dom(N)$ *satisfies* a functional dependency $X \to Y$ on N if and only if $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ holds whenever $\pi^N_X(t_1) = \pi^N_X(t_2)$ for any $t_1, t_2 \in r$ holds.

A *multi-valued dependency on* N is an expression of the form $X \twoheadrightarrow Y$ where $X, Y \in Sub(N)$. A finite set $r \subseteq Dom(N)$ *satisfies* a multi-valued dependency $X \twoheadrightarrow Y$ on N if and only if for all values $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ there is a value $t \in r$ with $\pi^N_{X \sqcup Y}(t) = \pi^N_{X \sqcup Y}(t_1)$ and $\pi^N_{X \sqcup Y^C}(t) = \pi^N_{X \sqcup Y^C}(t_2)$. □

Example 4.2 *Consider* $N =$*Pubcrawl(Person,Visit[Drink(Beer,Pub)]) and a typical snapshot* $r \subseteq Dom(N)$*:*

{ *(Sven,[(Lübzer,Deanos),(Kindl,Highflyers)]),*
(Sven,[(Kindl,Deanos),(Lübzer,Highflyers)]),
(Klaus-Dieter,[(Guiness,Irish Pub),(Speights,3Bar),(Guiness,Irish Pub)]),
(Klaus-Dieter,[(Kölsch,Irish Pub),(Bönnsch,3Bar),(Guiness,Irish Pub)]),
(Klaus-Dieter,[(Guiness,Highflyers),(Speights,Deanos),(Guiness,3Bar)]),

(Klaus-Dieter,[(Kölsch,Highflyers),(Bönnsch,Deanos),(Guiness,3Bar)]),
(Sebastian,[ ]) }

Obviously, the FD

$$Pubcrawl(Person) \rightarrow \ Pubcrawl(Visit[Drink(Pub)])$$

is not satisfied by r, neither is the FD

$$Pubcrawl(Person) \rightarrow \ Pubcrawl(Visit[Drink(Beer)]).$$

However, $\models_r Pubcrawl(Person) \twoheadrightarrow Pubcrawl(Visit[Drink(Pub)])$. This MVD informally says that a person has prefered lists of pubs, e.g. according to the weekday, and prefered lists of beers, e.g. according to the mood that person is in. Since a weekday is independent from the mood of a person, all possible combinations of these lists can occur. Note that

$$\models_r Pubcrawl(Person) \rightarrow Pubcrawl(Visit[\lambda])$$

holds. This means informally that the person determines the number of bars visited by that person. □

The notions of implication ($\models$) and derivability ($\vdash_{\mathfrak{R}}$) with respect to a set $\mathfrak{R}$ of inference rules for a class $\mathcal{C}$ of dependencies can be defined analogously to the notions in the RDM (see for instance [2, pp. 164-168]). Since (real-life) databases are always finite, implication is considered to be finite implication only in this article. Let Σ be a set of dependencies from $\mathcal{C}$ on some nested attribute N. We are interested in the set of all dependencies in $\mathcal{C}$ implied by Σ, i.e., $\Sigma^*_{\mathcal{C}} = \{\varphi \in \mathcal{C} \mid \Sigma \models \varphi\}$. Our aim is finding sets $\mathfrak{R}$ of inference rules which are sound ($\Sigma^+_{\mathcal{C}} \subseteq \Sigma^*_{\mathcal{C}}$) and complete ($\Sigma^*_{\mathcal{C}} \subseteq \Sigma^+_{\mathcal{C}}$) for the implication of dependencies in the class $\mathcal{C}$, and where $\Sigma^+_{\mathcal{C}} = \{\varphi \in \mathcal{C} \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$. In this paper, $\mathcal{C}$ will be the class of FDs and MVDs.

A dependency σ on some nested attribute N is called trivial if and only if $\models_r \sigma$ for every $r \in Dom(N)$. Trivial FDs and trivial MVDs have been characterised in [29].

Lemma 4.3 *Let $N \in \mathcal{N}A$, $X \rightarrow Y$ an FD on N and $X \twoheadrightarrow Y$ an MVD on N. Then is $X \rightarrow Y$ trivial if and only if $Y \leq X$ holds.Furthermore is $X \twoheadrightarrow Y$ trivial if and only if $Y \leq X$ or $X \sqcup Y = N$ holds.* □

Fagin proves in [23] that MVDs "provide a necessary and sufficient condition for a relation to be decomposable into two of its projections without loss of information (in the sense that the original relation is guaranteed to be the join of the two projections)." Let $N \in \mathcal{N}$A and $X, Y \in Sub(N)$. Let

$r_1 \subseteq Dom(X)$ and $r_2 \subseteq Dom(Y)$. Then

$$r_1 \bowtie r_2 \quad = \quad \{t \in Dom(X \sqcup Y) \mid \text{ there are } t_1 \in r_1, t_2 \in r_2 \text{ with } \pi_X^{X \sqcup Y}(t) = t_1 \text{ and } \pi_Y^{X \sqcup Y}(t) = t_2\}.$$

is called the *generalised join* $r_1 \bowtie r_2$ *of* r_1 and r_2. The *projection* $\pi_X(r)$ of $r \subseteq Dom(N)$ on $X \leq N$ is defined as $\{\pi_X^N(r) \mid t \in r\}$. In this sense, $r \subseteq Dom(N)$ satisfies the MVD $X \twoheadrightarrow Y$ exactly when r is the lossless generalised join of its projections on $X \sqcup Y$ and $X \sqcup Y^{\mathcal{C}}$, i.e., $r = \pi_{X \sqcup Y}(r) \bowtie \pi_{X \sqcup Y^C}(r)$. The following theorem has been proven in [29].

Theorem 4.4 *Let $N \in \mathcal{N}A$, $r \subseteq Dom(N)$ and $X \twoheadrightarrow Y$ an MVD on N. Then is $X \twoheadrightarrow Y$ satisified by r if and only if $r = \pi_{X \sqcup Y}(r) \bowtie \pi_{X \sqcup Y^C}(r)$.* □

Note that if an FD $X \twoheadrightarrow Y$ on N is satisfied by some $r \subseteq Dom(N)$, then $r = \pi_{X \sqcup Y}(r) \bowtie \pi_{X \sqcup Y^C}(r)$, but not vice versa. Take for instance $N = L(A, B)$ and $r = \{(a, b_1), (a, b_2)\}$. Then $\not\models_r L(A) \rightarrow L(B)$, but $r = \{a\} \bowtie \{b_1, b_2\}$.

Example 4.5 *In view of Theorem 4.4,*

$$\models_r \mathit{Pubcrawl(Person)} \twoheadrightarrow \mathit{Pubcrawl(Visit[Drink(Pub)])}$$

in Example 4.2 suggests to decompose r into the two projections $r_{Pubcrawl(Person, Visit[Drink(Pub)])}$ and $r_{Pubcrawl(Person, Visit[Drink(Beer)])}$, i.e.,

{(Sven,[Lübzer,Kindl]),(Sven,[Kindl,Lübzer]),(Klaus-Dieter,[Guiness,Speights,Guiness]),(Klaus-Dieter,[Kölsch,Bönnsch,Guiness]),(Sebastian,[ ]) }

and

{(Sven,[Deanos,Highflyers]),(Klaus-Dieter,[Irish Pub,3Bar,Irish Pub]),(Klaus-Dieter,[Highflyers,Deanos,3Bar]),(Sebastian,[ ]) },

respectively. □

4.1 Completeness

A sound and complete set of inference rules for FDs and MVDs has been provided in [9]. Natural extensions of the (sound and complete) rules from [36, p.80,81] are also sound in the presence of base, record and finite list types. Apart from these rules there is a further sound rule which allows to derive a non-trivial FD $X \rightarrow Y \sqcap Y^{\mathcal{C}}$ from an MVD $X \twoheadrightarrow Y$. The following theorem was the main result of [29].

Theorem 4.6 *The following set of inference rules*

$$\frac{}{X \to Y}\ {\scriptstyle Y \leq X} \qquad \frac{X \to Y}{X \to X \sqcup Y}$$

(reflexivity axiom) *(extension rule)*

$$\frac{X \to Y,\ Y \to Z}{X \to Z} \qquad \frac{X \to Y}{X \twoheadrightarrow Y}$$

(transitivity rule) *(implication rule)*

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow Y^{\mathcal{C}}} \qquad \frac{X \twoheadrightarrow Y}{W \sqcup X \twoheadrightarrow V \sqcup Y}\ {\scriptstyle V \leq W}$$

(Brouwerian-complement rule) *(multi-valued augmentation rule)*

$$\frac{X \twoheadrightarrow Y \quad Y \to Z}{X \twoheadrightarrow (Z \dot{-} Y)} \qquad \frac{X \twoheadrightarrow Y \quad Y \to Z}{X \to (Z \dot{-} Y)}$$

(pseudo-transitivity rule) *(mixed pseudo-transitivity rule)*

$$\frac{X \twoheadrightarrow Y \quad X \twoheadrightarrow Z}{X \twoheadrightarrow (Y \sqcup Z)} \qquad \frac{X \twoheadrightarrow Y \quad X \twoheadrightarrow Z}{X \twoheadrightarrow (Z \dot{-} Y)}$$

(multi-valued join rule) *(pseudo-difference rule)*

$$\frac{X \twoheadrightarrow Y}{X \to Y \sqcap Y^{\mathcal{C}}} \qquad \frac{X \twoheadrightarrow Y \quad X \twoheadrightarrow Z}{X \twoheadrightarrow (Y \sqcap Z)}$$

(mixed meet rule) *(multi-valued meet rule)*

is sound and complete for the implication of FDs and MVDs on a nested attribute N in the presence of base, record and finite list types. □

Note that reflexivity axiom, extension rule and transitivity rule form a sound and complete set of inference rules for the implication of FDs in the presence of base, record and finite list types (see [29]).

It is easy to see that all rules from Theorem 4.6 except the mixed meet rule are natural extensions of rules in the RDM (compare [36, p. 80,81]). Interpreting the mixed meet rule in relational databases means that the trivial FD $X \to \emptyset$ can be derived from the MVD $X \twoheadrightarrow Y$, and is therefore not needed.

4.2 Dependency Basis and Possessed Attributes

Definition 4.7 Let $N \in \mathcal{N}\mathrm{A}$. The *subattribute basis* $SubB(N)$ *of* N is the smallest set $SubB(N) \subseteq Sub(N)$ such that for all $X \in Sub(N)$ we have $X = \sqcup Z$ for some $Z \subseteq SubB(N)$. Every $X \in SubB(N)$ is called a *basis*

attribute for N. A basis attribute $X \in SubB(N)$ is called *maximal* if and only if $X \leq Y$ for some basis attribute $Y \in SubB(N)$ implies that $X = Y$ holds. Basis attributes that are not maximal are called *non-maximal*. Let $MaxB(N)$ denote the set of maximal basis attributes, and *non-MaxB(N)* the set of non-maximal basis attributes. □

Note that $\lambda \notin SubB(N)$ since $\lambda = \sqcup\emptyset$, and $SubB(N)$ is not an anti-chain with respect to $\leq$. It is true that $X = X^{\mathcal{CC}} \sqcup (X \sqcap X^{\mathcal{C}})$ holds in every Brouwerian Algebra. A basis attribute $Y \in SubB(N)$ is maximal if and only if $Y = Y^{\mathcal{CC}}$ holds, and non-maximal if and only if $Y = Y \sqcap Y^{\mathcal{C}}$ holds.

Example 4.8 *Let $N = A(B, C[D(E, F[G])])$. The subattribute basis is then*

$$SubB(N) = \{A(B), A(C[\lambda]), A(C[D(F[\lambda])]), A(C[D(E)]), A(C[D(F[G])])\}.$$

The maximal basis attributes are $A(B), A(C[D(E)])$ and $A(C[D(F[G])])$. The non-maximal basis attributes are $A(C[\lambda])$ and $A(C[D(F[\lambda])])$. □

Consider the set of all Y with $X \twoheadrightarrow Y \in \Sigma^+$ for a fixed X defined on some nested attribute N. According to the multi-valued join, multi-valued meet and pseudo-difference rule this set, partially ordered by $\leq$, forms a Brouwerian Algebra. Due to the mixed meet rule, all basis attributes of Y which are not maximal in N are already functionally determined by X.

Definition 4.9 Let $N \in \mathcal{N}\mathrm{A}$, $X \in Sub(N)$ and Σ a set of multi-valued and functional dependencies on N. Let $Dep(X)$ be the set of all $Y \in Sub(N)$ with $X \twoheadrightarrow Y \in \Sigma^+$ and $X^+ = \sqcup\{Y \mid X \rightarrow Y \in \Sigma^+\}$. Let $X^M \subseteq Sub(N)$ have the following properties:

(i) for all $U \in MaxB(N)$ there is a unique $V \in X^M$ with $U \leq V$,

(ii) for all $U \in X^M$ there is some $W \subseteq MaxB(N)$ with $U = \sqcup W$,

(iii) for all $V \in Dep(X)$ there is some $Z \subseteq X^M$ with $V^{\mathcal{CC}} = \sqcup Z$, and

(iv) X^M is maximal with these properties with respect to $\subseteq_{\mathrm{gen}}$.

The *dependency basis* of X with respect to Σ is $DepB(X) = SubB(X^+) \cup X^M$. □

Note that $\{MaxB(W) \mid W \in X^M\}$ is the partition of $MaxB(N)$ which is generated by $\{MaxB(Y^{\mathcal{CC}}) \mid Y \in Dep(X)\}$. The first property says that every maximal basis attribute of N is the subattribute of exactly one element in X^M. The second property guarantees that every element in X^M is the join of maximal basis attributes of N. If $X \twoheadrightarrow V \in \Sigma^+$ holds, then the join of all basis attributes in V which are maximal in N (, i.e. $V^{\mathcal{CC}}$,) is the join over elements of X^M by property three. Finally, the last property guarantees the uniqueness of the dependency basis and that $X \twoheadrightarrow W \in \Sigma^+$ holds for all

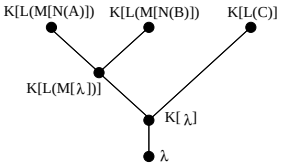

Fig. 2. The subattribute basis of $K[L(M[N(A,B)],C)]$

$W \in X^M$. An MVD $X \twoheadrightarrow Y$ is derivable from Σ iff the right-hand side Y is the join over some elements of the dependency basis of X with respect to Σ.

Proposition 4.10 *Let $N \in \mathcal{N}\mathrm{A}$ and Σ as set of FDs and MVDs on N. Then*

(i) $X \twoheadrightarrow Y \in \Sigma^+$ *if and only if* $Y = \sqcup Z$ *for some* $Z \subseteq DepB(X)$

(ii) $X \to Y \in \Sigma^+$ *if and only if* $Y \leq X^+$. □

Proving the completeness result for FDs and MVDs involves the definition of some instance which satisfies all dependencies in Σ. This instance initially contains two elements t_1, t_2 which are coincident on exactly all attributes which are functionally determind by some fixed X. Afterwards new elements are generated and added to the instance by exhaustively combining values from t_1 on some $W \subseteq X^M$ and the values from t_2 on $X^M \backslash W$. Let $W, W' \in X^M$. Since the meet $W \sqcap W'$ is not necessarily equal to λ one needs to show that such a construction is possible in general. It will turn out that $SubB(W \sqcap W')$ contains only attributes already functionally determined by X.

Definition 4.11 Let $N \in \mathcal{N}\mathrm{A}$, $X' \subseteq MaxB(N)$ and $X = \sqcup X'$. A basis attribute $Y \in SubB(X)$ is *possessed by* X if and only if every basis attribute $Z \in SubB(N)$ with $Y \leq Z$ is also a subattribute of X ($Z \leq X$). □

It follows that $SubB(W \sqcap W')$ with $W, W' \in X^M$ contains only basis attributes of W or W' which are neither possessed by W nor by W'.

Example 4.12 *Let $K[L(M[N(A,B)],C)] \in \mathcal{N}A$, and $X = K[L(M[N(A,B)])]$. Then X does possess $K[L(M[\lambda])]$, but does not possess $K[\lambda]$. For an illustration see also Figure 2.*

A basis attribute is not possessed by some X exactly if it is also a basis attribute of $X^{\mathcal{C}}$. According to the mixed meet rule it follows that basis attributes which are not possessed by some element in X^M are functionally determined by X. Suppose $DepB(X) = SubB(X^+) \cup \{W_{0,1}, \ldots, W_{0,m}, W_1, \ldots, W_k\}$ with $W_{0,i} \leq X^+$ for $i = 1, \ldots, m$ and $W_1, \ldots, W_k \not\leq X^+$. We have seen that $SubB(W_i \sqcap W_j)$, $i \neq j$, contains only basis attributes of W_i or W_j neither possessed by W_i nor by W_j. It follows that $X \to W_i \sqcap W_j$ holds.

In summary, the construction is based on the relational theory for maximal basis attributes and the fact that non-maximal basis attributes not possessed

by any $W \in X^M$ are functionally determined by X.

5 Computing the Dependency Basis

Given a set Σ of dependencies from one of our classes, and a further dependency σ from the same class, the membership problem is to decide whether $\Sigma \models \sigma$ holds. In view of Theorem 4.6, the membership problem is decidable for FDs and MVDs. That is, given some Σ, we can obviously enumerate all the dependencies that can be derived from it. However, the enumeration algorithm is time consuming and therefore impractical. We will now present an efficient membership algorithm for FDs and MVDs. Our main objective is the presentation of the algorithm, argue that it works correctly and in polynomial time. Implementation details are not discussed. Thus the time bound will only be a rough estimate of the upper bound.

Algorithm 5.1 (Attribute Set Closure and Dependency Basis)
Input: $\boldsymbol{N} \in \boldsymbol{N}$A, $\boldsymbol{X} \in \boldsymbol{Sub(N)}$, set $\boldsymbol{\Sigma}$ of FDs and MVDs on $\boldsymbol{N}$
Output: $\boldsymbol{X}^+_{\text{alg}}$ and $\text{DepB}_{\text{alg}}(\boldsymbol{X})$
Method:
VAR $DB_{\text{new}}, DB_{\text{old}} \subseteq MaxB(N)$, $X_{\text{new}}, X_{\text{old}}, W, \overline{U}, \widetilde{V}, U' \in Sub(N)$;

$X_{\text{new}} := X$;
$DB_{\text{new}} := MaxB(X^{\mathcal{CC}}) \cup \{X^{\mathcal{C}}\}$;
REPEAT
  $X_{\text{old}} := X_{\text{new}}$;
  $DB_{\text{old}} := DB_{\text{new}}$;
  FOR each $U \to V \in \Sigma$ DO
    $\overline{U} := \sqcup\{W \in DB_{\text{new}} \mid \exists U'.U' \text{ possessed by } W, U' \not\leq X_{\text{new}}, U' \leq U\}$;
    $\widetilde{V} := V \dot{-} \overline{U}$;
    IF $\widetilde{V} \neq \lambda$ THEN BEGIN
      $X_{\text{new}} := X_{\text{new}} \sqcup \widetilde{V}$;
      $DB_{\text{new}} := \{(W \dot{-} \widetilde{V})^{\mathcal{CC}} \mid W \in DB_{\text{new}}, (W \dot{-} \widetilde{V})^{\mathcal{CC}} \neq \lambda\} \cup MaxB(\widetilde{V}^{\mathcal{CC}})$;
    END;
  ENDDO;
  FOR each $U \twoheadrightarrow V \in \Sigma$ DO
    $\overline{U} := \sqcup\{W \in DB_{\text{new}} \mid \exists U'.U' \text{ possessed by } W, U' \not\leq X_{\text{new}}, U' \leq U\}$;
    $\widetilde{V} := V \dot{-} \overline{U}$;
    IF $\widetilde{V} \neq \lambda$ THEN BEGIN
      $X_{\text{new}} := X_{\text{new}} \sqcup (\widetilde{V} \sqcap \widetilde{V}^{\mathcal{C}})$;

FOR each $W \in DB_{\text{new}}$ DO
IF $(\widetilde{V} \sqcap W)^{\mathcal{CC}} \neq \lambda$ AND $(\widetilde{V} \sqcap W)^{\mathcal{CC}} \neq W$ THEN
$DB_{\text{new}} := (DB_{\text{new}} - \{W\}) \cup \{(\widetilde{V} \sqcap W)^{\mathcal{CC}}, (W \dot{-} \widetilde{V})^{\mathcal{CC}}\}$;
ENDDO;
END;
ENDDO;
UNTIL $(X_{\text{new}} = X_{\text{old}})$ AND $(DB_{\text{new}} = DB_{\text{old}})$;
$X^{+}_{\text{alg}} := X_{\text{new}}$;
$DepB_{\text{alg}}(X) := SubB(X^{+}_{\text{alg}}) \cup DB_{\text{new}}$;
RETURN$(X^{+}_{\text{alg}}, DepB_{\text{alg}}(X))$;

□

In order to become more familiar with Algorithm 5.1 we present a simple example.

Example 5.1 *Suppose the input for Algorithm 5.1 is as follows:*

- $N = L_1(L_2[L_3[L_4(A, B, C)]], L_5[L_6(D, E)],\ L_7(F,\ L_8[L_9(G,\ L_{10}[H])], I))$,
- $U_1 = L_1(L_5[\lambda], L_7(F, L_8[L_9(G)], I))$, $V_1 = L_1(L_2[L_3[L_4(C)]], L_5[L_6(E)])]$,
- $U_2 = L_1(L_2[L_3[\lambda]], L_7(F))$, $V_2 = L_1(L_2[L_3[L_4(A)]], L_7(L_8[L_9(G)], I))$,
- $U_3 = L_1(L_7(F, L_8[L_9(L_{10}[\lambda])]))$, $V_3 = L_1(L_2[L_3[\lambda]], L_5[L_6(D)])$,
- $\Sigma = \{U_1 \twoheadrightarrow V_1, U_2 \rightarrow V_2, U_3 \twoheadrightarrow V_3\}$ *and* $X = L_1(L_7(F, L_8[L_9(L_{10}[H])]))$.

After the initialization we have

- $X_{new} = X$
- $DB_{new} = \{L_1(L_2[L_3[L_4(A, B, C)]], L_5[L_6(D, E)], L_7(L_8[L_9(G)], I))$; $L_1(L_7(F))$; $L_1(L_7(L_8[L_9(L_{10}[H])]))\}$.

The initial state is illustrated in Figure 3. Functionally determined basis attributes are marked with a circle, remaining maximal basis attributes are boxed according to their membership.

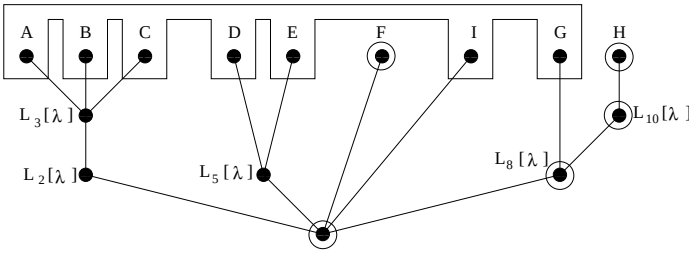

Fig. 3. Initialisation for $DepB_{\text{alg}}(X)$

The first pass through the REPEAT UNTIL loop yields the following in-

termediate results:

(i) $U_2 \rightarrow V_2$:
$\overline{U} = L_1(L_2[L_3[L_4(A,B,C)]], L_5[L_6(D,E)],\ L_7(L_8[L_9(G)], I))$,
$\widetilde{V} = \lambda$ *and therefore no changes*

(ii) $U_1 \twoheadrightarrow V_1$:
$\overline{U} = L_1(L_2[L_3[L_4(A,B,C)]], L_5[L_6(D,E)],\ L_7(L_8[L_9(G)], I))$,
$\widetilde{V} = \lambda$ *and therefore no changes*

(iii) $U_3 \twoheadrightarrow V_3$:
$\overline{U} = \lambda$, $\widetilde{V} = V_3$
$X_{new} = L_1(L_2[L_3[\lambda]], L_5[\lambda], L_7(F, L_8[L_9(L_{10}[H])]))$
$DB_{new} = \{L_1(L_2[L_3[L_4(A,B,C)]], L_5[L_6(E)], L_7(L_8[L_9(G)], I));\ L_1(L_7(F));$
$L_1(L_7(L_8[L_9(L_{10}[H])]));\ L_1(L_5[L_6(D)])\}$

The second pass through the REPEAT UNTIL loop yields the following intermediate results:

(i) $U_2 \rightarrow V_2$:
$\overline{U} = \lambda$, $\widetilde{V} = V_2$,
$X_{new} = L_1(L_2[L_3[L_4(A)]], L_5[\lambda], L_7(F, L_8[L_9(G,\ L_{10}[H])], I))$,
$DB_{new} = \{L_1(L_2[L_3[L_4(A)]]);\ L_1(L_7(L_8[L_9(G)]));\ L_1(L_7(I));$
$L_1(L_2[L_3[L_4(B,C)]], L_5[L_6(E)]);\ L_1(L_7(F));\ L_1(L_7(L_8[L_9(L_{10}[H])]));$
$L_1(L_5[L_6(D)])\}$

(ii) $U_1 \twoheadrightarrow V_1$:
$\overline{U} = \lambda$, $\widetilde{V} = V_1$,
$X_{new} = L_1(L_2[L_3[L_4(A)]], L_5[\lambda], L_7(F, L_8[L_9(G,\ L_{10}[H])], I))$,
$DB_{new} = \{L_1(L_2[L_3[L_4(A)]]);\ L_1(L_7(L_8[L_9(G)]));$
$L_1(L_7(I));\ L_1(L_2[L_3[L_4(B)]]);\ L_1(L_2[L_3[L_4(C)]], L_5[L_6(E)]);\ L_1(L_7(F));$
$L_1(L_7(L_8[L_9(L_{10}[H])]));\ L_1(L_5[L_6(D)])\}$

(iii) $U_3 \twoheadrightarrow V_3$:
$\overline{U} = \lambda$, $\widetilde{V} = V_3$ *and therefore no changes.*

The next pass through the REPEAT UNTIL loop yields nothing new. We therefore have the output

- $X^+_{alg} = L_1(L_2[L_3[L_4(A)]], L_5[\lambda], L_7(F, L_8[L_9(G,\ L_{10}[H])], I))$ *and*
- $DepB_{alg}(X) = \{L_1(L_2[\lambda]);\ L_1(L_2[L_3[\lambda]]);\ L_1(L_2[L_3[L_4(A)]]);\ L_1(L_5[\lambda]);$
$L_1(L_7(F));\ L_1(L_7(L_8[\lambda]));\ L_1(L_7(L_8[L_9(G)]));\ L_1(L_7(L_8[L_9(L_{10}[\lambda])]));$
$L_1(L_7(L_8[L_9(L_{10}[H])]));\ L_1(L_7(I));\ L_1(L_5[L_6(D)]);\ L_1(L_2[L_3[L_4(B)]]);$
$L_1(L_2[L_3[L_4(C)]], L_5[L_6(E)])\}$.

Figure 4 illustrates $DepB_{alg}(X)$. □

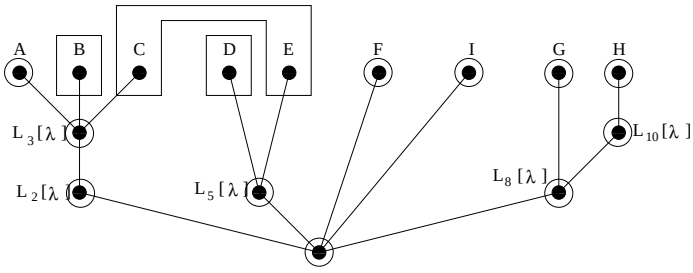

Fig. 4. Final State for $DepB_{\mathrm{alg}}(X)$ from Example 5.1.

6 Correctness and Complexity

The correctness proof of Algorithm 5.1 is sketched, i.e., $X^+_{\mathrm{alg}} = X^+$ and $DepB_{\mathrm{alg}}(X) = DepB(X)$. The first step is to show that the output of the algorithm admits only dependencies which can also be inferred.

Lemma 6.1 *Let $N \in \mathcal{N}A$, $X \leq N$, Σ a set of FDs and MVDs on N and $DepB_{alg}(X) = SubB(X^+_{alg}) \cup X^M_{alg}$ the output of Algorithm 5.1. Then*

- $X \twoheadrightarrow W_j \in \Sigma^+$ *for all* $W_j \in DepB_{alg}(X)$ *and*
- $X \rightarrow X^+_{alg} \in \Sigma^+$ *hold.*

Proof. [sketch] The proof can be done by induction on the number of passes through the two outermost FOR loops of Algorithm 5.1. □

Consider the proper chain $\Sigma = \Sigma^0 \subset \Sigma^1 \subset \Sigma^n = \Sigma^+$ where Σ^i results from Σ^{i-1} by adding exactly one functional or multi-valued dependency which is not in Σ^{i-1} and can be derived by applying one of the inference rules from Theorem 4.6 to dependencies in Σ^{i-1}. On the way to showing the correctness of Algorithm 5.1 we have to justify that it is sufficient to consider FDs and MVDs in Σ. That is, we need to show that dependencies in $\Sigma^+ \backslash \Sigma$ do not alter the dependency basis. Suppose the algorithm does not only select $U \rightarrow V, U \twoheadrightarrow V \in \Sigma$ within the two FOR loops, but all FDs and MVDs from some fixed Σ^i instead. Denoting the respective output by $(X^+_{\mathrm{alg},i}, DepB_{\mathrm{alg},i}(X))$, we define

$$\begin{aligned}(\Sigma^i)^+_{\mathrm{alg}} \quad = \quad & \{X \rightarrow Y \mid Y \leq X^+_{\mathrm{alg},i}\} \cup \\ & \{X \twoheadrightarrow Y \mid Y = \sqcup Z \text{ for some } Z \subseteq DepB_{\mathrm{alg},i}(X)\}.\end{aligned}$$

Then it is obvious that $\Sigma^+_{\mathrm{alg}} = (\Sigma^0)^+_{\mathrm{alg}} \subseteq (\Sigma^1)^+_{\mathrm{alg}} \subseteq \cdots \subseteq (\Sigma^n)^+_{\mathrm{alg}}$ holds. The algorithm is designed in such a way that $\Sigma^i \subseteq (\Sigma^i)^+_{\mathrm{alg}}$ holds for any i. Furthermore, it can be shown that $\Sigma^+_{\mathrm{alg}} = (\Sigma^1)^+_{\mathrm{alg}}$. It follows then immediately

that $\Sigma^+ \subseteq (\Sigma^n)^+_{\text{alg}} = \Sigma^+_{\text{alg}}$ holds. Since also $\Sigma^+_{\text{alg}} \subseteq \Sigma^+$ by Lemma 6.1 and Proposition 4.10, we have indeed shown that $\Sigma^+_{\text{alg}} = \Sigma^+$.

Lemma 6.2 $\Sigma^+_{alg} = (\Sigma^1)^+_{alg}$

Proof. [sketch] Let $N \in \mathcal{N}\text{A}$, $X \leq N$ and Σ a set of FDs and MVDs on N. One can show that the functional or multi-valued dependency in $\Sigma^1 \backslash \Sigma$ does not affect the output $(X^+_{\text{alg}}, DepB_{\text{alg}}(X))$ of Algorithm 5.1. Therefore, every inference rule from Theorem 4.6 needs to examined in turn. □

Putting everything together we obtain the following result.

Theorem 6.3 *Let $N \in \mathcal{N}A$, $X \leq N$ and Σ a set of FDs and MVDs on N. Then Algorithm 5.1 always terminates and computes attribute set closure X^+ and dependency basis $DepB(X)$ for X with respect to Σ.*

Proof. It remains to show the termination of Algorithm 5.1. After the initialization and after each pass through the REPEAT UNTIL loop, the set $MaxBasis = \{MaxB(Z) \mid Z \in DB_{\text{new}}\}$ is a partition of $MaxB(N)$. Consequently, the number of sets in any such partition is at most $\mid MaxB(N) \mid$, the number of maximal basis attributes of N. However, after each pass through the REPEAT UNTIL loop (except the last) the partition $MaxBasis$ is refined, and the number of sets in it increases, or the number of elements in $non\text{-}MaxB(X_{\text{new}})$ increases. It follows, that the REPEAT UNTIL loop is executed at most $\mid SubB(N) \mid = \mid MaxB(N) \mid + \mid non\text{-}MaxB(N) \mid$ times, and therefore the algorithm terminates. □

We will now show that the membership problem of FDs and MVDs can be decided in polynomial time. When studying the time complexity of Algorithm 5.1 we consider nested attributes N as sets of attributes, i.e., instead of looking at N, we rather use $SubB(N)$. Let $N \in \mathcal{N}\text{A}$, $X \in Sub(N)$ and Σ a set of FDs and MVDs on N be the input for Algorithm 5.1. We use $\mid N \mid$ to denote the size of N, that is $\mid N \mid$ is the number of basis attributes of N, i.e., $\mid SubB(N) \mid$. It is relatively easy to see that $SubB(X)$ and $MaxB(X)$ can be computed in time $\mathcal{O}(\mid N \mid)$. Furthermore, the union and intersection of sets can be computed in time $\mathcal{O}(\mid N \mid)$ as well. Since $SubB(X \sqcup Y) = SubB(X) \cup SubB(Y)$ and $SubB(X \sqcap Y) = SubB(X) \cap SubB(Y)$, join and meet operation are linear in $\mid N \mid$, as well. The pseudo-difference, and therefore the Brouwerian-complement operation as well, can be implemented in quadratic time:

$SubB(X \dot{-} Y) := SubB(X)$;
FOR ALL $A \in SubB(X)$ DO
  IF $A \in SubB(Y)$ THEN $SubB(X \dot{-} Y) := SubB(X \dot{-} Y) - \{A\}$;

ENDDO;
FOR ALL $A \in SubB(X \dot{-} Y)$ DO
  $SubB(X \dot{-} Y) := SubB(X \dot{-} Y) \cup SubB(A)$;
ENDDO;

, i.e., time $\mathcal{O}(\mid N \mid^2)$. Next, we will write down an algorithm which computes $\overline{U} := \sqcup\{W \in DB_{\text{new}} \mid \exists U'.U' \text{ is possessed by } W, U' \not\leq X_{\text{new}}, U' \leq U\}$. Recall that U' is possessed by some $W \in DB_{\text{new}}$ if and only if $U' \in SubB(W)$ and $U' \notin SubB(W^{\mathcal{C}})$.

$\overline{U} := \lambda$;
WHILE $(SubB(U) \neq \emptyset)$ AND $(DB_{\text{new}} \neq \emptyset)$ DO
  SELECT $U' \in SubB(U)$;
  $SubB(U) := SubB(U) - \{U'\}$;
  IF $U' \notin SubB(X_{\text{new}})$ THEN
    FOR ALL $W \in DB_{\text{new}}$ DO
      IF $(U' \in SubB(W))$ AND $(U' \notin SubB(W^{\mathcal{C}}))$ THEN
        $\overline{U} := \overline{U} \sqcup W$;
        $DB_{\text{new}} := DB_{\text{new}} - \{W\}$;
      ENDIF;
    ENDDO;
  ENDIF;
ENDDO;

This demonstrates that $\overline{U}$ can be computed in time $\mathcal{O}(\mid N \mid^3)$. Let us now look at the time complexity to refine X_{new} and DB_{new}, respectively. First, consider the case where this refinement has been triggered by a functional dependency $U \to V \in \Sigma$. If $\widetilde{V} \neq \lambda$, then $X_{\text{new}} := X_{\text{new}} \sqcup \widetilde{V}$ which takes time in $\mathcal{O}(\mid N \mid)$. In order to compute $DB_{\text{new}} := \{(W \dot{-} \widetilde{V})^{\mathcal{CC}} \mid W \in DB_{\text{new}}, (W \dot{-} \widetilde{V})^{\mathcal{CC}} \neq \lambda\} \cup MaxB(\widetilde{V}^{\mathcal{CC}})$, we need to compute $(W \dot{-} \widetilde{V})^{\mathcal{CC}}$ in time $\mathcal{O}(\mid N \mid^2)$ for every $W \in DB_{\text{new}}$. Since DB_{new} has at most $\mid MaxB(N) \mid$ elements, this takes time in $\mathcal{O}(\mid N \mid^3)$. Computing $MaxB(\widetilde{V}^{\mathcal{CC}})$ and forming the union is in $\mathcal{O}(\mid N \mid^2)$. Consider now the case where the refinement is triggered by some multi-valued dependency $U \twoheadrightarrow V \in \Sigma$. If $\widetilde{V} \neq \lambda$, then $X_{\text{new}} := X_{\text{new}} \sqcup (\widetilde{V} \sqcap \widetilde{V}^{\mathcal{C}})$ which takes time in $\mathcal{O}(\mid N \mid^2)$. As the computation of $(\widetilde{V} \sqcap W)^{\mathcal{CC}}$ and $(W \dot{-} \widetilde{V})^{\mathcal{CC}}$ takes time in $\mathcal{O}(\mid N \mid^2)$, the inner FOR loop for the refinement of DB_{new} takes $\mathcal{O}(\mid N \mid^3)$ steps.

It follows that each pass through the REPEAT UNTIL loop takes time in $\mathcal{O}(\mid N \mid^3 \cdot \mid \Sigma \mid)$. As we have seen before, the REPEAT UNTIL loop is executed at most $\mid N \mid$ times. Therefore, the time complexity of Algorithm

5.1 is $\mathcal{O}(\mid N \mid^4 \cdot \mid \Sigma \mid)$.

Theorem 6.4 *Let $N \in \mathcal{N}A$, Σ a set of FDs and MVDs on N and σ an FD or MVD on N. The membership problem whether $\Sigma \models \sigma$ holds can be decided in time $\mathcal{O}(\mid N \mid^4 \cdot \mid \Sigma \mid)$.*

Proof. Let σ be the FD $X \to Y$. Algorithm 5.1 computes the attribute set closure X^+ in time $\mathcal{O}(\mid N \mid^4 \cdot \mid \Sigma \mid)$. It follows that $\Sigma \models X \to Y$ if and only if $Y \leq X^+$ according to Proposition 4.10. To decide whether $Y \leq X^+$ holds takes time in $\mathcal{O}(\mid N \mid)$.

Let σ be the MVD $X \twoheadrightarrow Y$. Algorithm 5.1 computes the dependency basis $DepB(X) = SubB(X^+) \cup X^M$ in time $\mathcal{O}(\mid N \mid^4 \cdot \mid \Sigma \mid)$. It follows that $\Sigma \models X \twoheadrightarrow Y$ if and only if $Y = \sqcup Z$ for some $Z \subseteq DepB(X)$ according to Proposition 4.10. To decide whether Y is the join of some elements in $DepB(X)$ takes time in $\mathcal{O}(\mid N \mid^2)$. This proves the theorem. □

7 Conclusion and Future Work

FDs and MVDs have proven significance for the logical design of relational databases (see [2,7,8,13,15,16,21,22,23,24,36,41,45]). We have continued studying the class of FDs and MVDs in databases supporting base, record and finite list types. In [29] a sound and complete set of inference rules for the implication of FDs and MVDs has been presented, generalising the set of inference rules from [9] for the RDM. Based on this axiomatisation a membership algorithm, deciding the finite implication problem for FDs and MVDs with base, record and finite list types, has been presented in this article. This membership algorithm is a natural extension of Beeri's algorithm from [6] for relational databases. It was argued that the algorithm works correctly and runs in polynomial time. This shows that the membership problem can still be solved efficiently when complex data types are present in databases. The results also show that the approach based on types and nested attributes pays off for the development of a unified dependency theory for advanced data models.

Derivations not using the Brouwerian-complement rule are of particular interest, see for instance [6] for an explanation why the complementation rule is of particular interest in the RDM. In the relational case it is possible to decide in polynomial time whether a given FD or MVD can be derived from a given set of FDs and MVDs without using the complementation rule. We are confident that this decision procedure can be extended to databases supporting various types.

The inference rules from Theorem 4.6 are expected to be redundant. A detailed study of minimal sets of inference rules, in the sense that any proper

subset will not be complete anymore, was outside the scope of this paper. It might help to simplify the lengthy correctness proof for Algorithm 5.1. Results similar to [14,32] are expected.

There are many ways of continuing our research. For the future, we would like to explore richer type systems containing sets, multisets, unions and even references leading to rational trees. The class of FDs in the presence of base, record and finite set types has already been studied in [27] and has led to a more sophisticated set of inference rules since the extension rule is no longer valid in the presence of sets. We have also looked at extending this result to lists and multisets. FDs can be quite easily captured in the presence of base, record and finite list types only. On the other hand, MVDs show an interesting behaviour in the presence of finite set types, in the sense that Theorem 4.4 is no longer valid. That is MVDs deviate from binary join dependencies.

The main objective is of course the study of normal forms for nested attributes which guarantee well-designed databases. The desirable goal would be a theory extending the results from [45] to databases supporting various types. The start for this research has already been made in [27] where the Higher Level normal form has been proposed as a strictly weaker normal form than Boyce-Codd normal form. For the class of FDs in the presence of base, record and finite set types, this normal form has been proven equivalent to the absence of redundancies and sufficient for the absence of abnormal update behaviour. We would like to generalise the fourth normal form on the basis of several type systems and semantically justify these generalisations. The membership problem presented in this article will then be very useful for eliminating redundancies.

In the future, various other classes of relational dependencies (see [41]), including join dependencies and inclusion dependencies, together with their interactions should be studied with respect to various type systems.

References

[1] Abiteboul, S., P. Buneman and D. Suciu, *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, 2000.

[2] Abiteboul, S., R. Hull and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995.

[3] Arenas, M. and L. Libkin, A normal form for XML documents. In *PODS 2002*. ACM, 2002.

[4] Armstrong, W. W., Dependency structures of database relationships. *Information Processing*, pages 580–583, 1974.

[5] Batini, C., S. Ceri and S. B. Navathe, *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin Cummings, 1992.

[6] Beeri, C., On the membership problem for functional and multivalued dependencies in relational databases. *Transactions on Database Systems*, 5(3):241–259, 1980.

[7] Beeri, C. and P. A. Bernstein, Computational problems related to the design of normal form relational schemata. In *Transactions on Database Systems*, pages 30–59. Association for Computing Machinery, 1979.

[8] Beeri, C., P. A. Bernstein and N. Goodman, A sophisticate's introduction to database normalization theory. In *Proceedings of 4th International Conference on Very Large Databases*, pages 113–124, 1978.

[9] Beeri, C., R. Fagin and J. H. Howard, A complete axiomatization for functional and multivalued dependencies in database relations. In *Proceedings of the International Conference on Management of Data*, pages 47–61. Association for Computing Machinery, 1977.

[10] Beeri, C., A. Mendelzon, Y. Sagiv and J. Ullman, Equivalence of relational database schemes. *SIAM Hournal on Computation*, 10:647–656, 1981.

[11] Beeri, C. and J. Rissanen, Faithful representation of relational database schemes. IBM Research Report, San Jose, 1980.

[12] Bernstein, P., Synthesizing third normal form relations from functional dependencies. *ACM Transactions on Database Systems*, 1:277–298, 1976.

[13] Bernstein, P. A. and N. Goodman, What does Boyce-Codd normal form do? In *Proceedings of the International Conference on Very Large Databases*, pages 245–259, 1980.

[14] Biskup, J., On the complementation rule for multivalued dependencies in database relations. *Acta Informatica*, 10(3):297–305, 1978.

[15] Biskup, J. Database schema design theory:achievements and challenges. In *Proceedings of the 6th International Conference on Information Systems and Management of Data*, number 1066 in Lecture Notes in Computer Science, pages 14–44. Springer, 1995.

[16] Biskup, J. Achievements of relational database schema design theory revisited. In *Semantics in databases*, number 1358 in Lecture Notes in Comptuer Science, pages 29–54. Springer, 1998.

[17] Bry, F. and P. Kröger, A computational biology database digest: data, data analysis, and data management. *Distributed and Parallel Databases*, 13(1):7–42, 2003.

[18] Chen, P. P., The entity-relationship model: Towards a unified view of data. *ACM Transactions Database Systems 1*, pages 9–36, 1976.

[19] Chen, P. P., English sentence structure and entity-relationship diagrams. *Information Science 29*, pages 127–149, 1983.

[20] Codd, E. F., A relational model of data for large shared data banks. *Communications of the ACM*, pages 377–387, 1970.

[21] Codd, E. F., Further normalization of the database relational model. In *Courant Computer Science Symposia 6: Data Base Systems*, pages 33–64. Prentice-Hall, 1972.

[22] Codd, E. F., Recent investigations in relational database system. In *Proceedings of the IFIP Conference*, pages 1017–1021, 1974.

[23] Fagin, R., Multivalued dependencies and a new normal form for relational databases. *Association for Computing Machinery*, 2(3):262–278, 1977.

[24] Fagin, R., A normal form for relational databases that is based on domains and keys. *Transactions on Database Systems*, pages 387–415, 1981.

[25] Gardarin, G., J.-P. Cheiney, G. Kiernan, D. Pastre and H. Stora, Managing complex objects in an extensible relational dbms. In *Proceedings of the Fifteenth International Conference on Very Large Data Bases, August 22-25, 1989, Amsterdam, The Netherlands*, pages 55–65. Morgan Kaufmann, 1989.

[26] Hartmann, S. and S. Link, More functional dependencies for XML. In *Advances in Databases and Information Systems*, volume 2798, pages 355–369. Springer Lecture Notes in Computer Science, 2003.

[27] Hartmann, S. and S. Link, On functional dependencies in advanced data models. *Electronic Notes in Theoretical Computer Science*, 84, 2003.

[28] Hartmann, S., S. Link and K.-D. Schewe, Generalizing Boyce-Codd normal form to conceptual databases. In *Pre-Proceedings of the 13th European-Japanese Conference on Information Modeling and Knowledge Bases*, pages 93–110, 2003.

[29] Hartmann, S., S. Link and K.-D. Schewe, Reasoning about functional and multi-valued dependencies in the presence of lists. In *Proceedings of Third International Symposium on Foundations of Information and Knowledge Systems, Vienna, Austria*. Springer Lecture Notes in Computer Science, 2004.

[30] Hull, R. and R. King, Semantic database modeling: Survey, applications and research issues. *ACM Computing Surveys*, 19(3), 1987.

[31] McKinsey, J. and A. Tarski, On closed elements in closure algebras. *Annals of Mathematics*, 47:122–146, 1946.

[32] Mendelzon, A., On axiomatising multivalued dependencies in relational databases. *Journal of the ACM*, 26(1):37–44, 1979.

[33] Mok, W. Y., Y. K. Ng and D. W. Embley, A normal form for precisely charachterizing redundancy in nested relations. *Transactions on Database Systems*, 21:77–106, 1996.

[34] Naqvi, S. and S. Tsur, *A logical language for data and knowledge bases*. Computer Science Press, 1989.

[35] Özsoyoglu, Z. M. and L. Y. Yuan, A new normal form for nested relations. *Transactions on Database Systems*, 12:111–136, 1987.

[36] Paredaens,J., P. De Bra, M. Gyssens and D. Van Gucht, *The Structure of the Relational Database Model*. Springer-Verlag, 1989.

[37] Richardson, J., Supporting lists in a datamodel. In *Proceeding of VLDB*, pages 127–192, 1992.

[38] Schewe, K. D. and B. Thalheim, Fundamental concepts of object oriented databases. *Acta Cybernetica*, 11(4):49–85, 1993.

[39] Seshadri, P., M. Livny and R. Ramakrishnan, The design and implementation of sequence database system. In *Proceedings of the Twentysecond International Conference on Very Large Data Bases, Mumbai, India*, 1996.

[40] Tari, Z., J. Stokes and S. Spaccapietra, Object normal forms and dependency constraints for object-oriented schemata. *ACM ToDS*, 22:513–569, 1997.

[41] Thalheim, B., *Dependencies in Relational Databases*. Teubner-Verlag, 1991.

[42] Thalheim, B. Foundations of entity-relationship modeling. *Annals of Mathematics and Artificial Intelligence*, 6:197–256, 1992.

[43] Thalheim, B., *Entity-Relationship Modeling: Foundations of Database Technology*. Springer-Verlag, 2000.

[44] Tjoa, A. M. and L. Berger, Transformation of requirement specifications expressed in natural language into an eer model. In *Entity-Relationship Approach*, volume 823. Springer Lecture Notes Series, 1993.

[45] Vincent, M., *The semantic justification for normal forms in relational database design*. PhD thesis, Monash University, Melbourne, Australia, 1994.

[46] Vincent, M., J. Liu and C. Liu, A redundancy free 4nf for XML. In *XML Database Symposium*, 2003.

[47] W3C. Xml schema part 2: Datatypes. http://www.w3.org/TR/xmlschema-2/#datatype, 2001.